

高级语言程序设计

第 1 章

程序设计和C/C++语言

华中师范大学物理学院 李安邦

高级语言程序设计

一、课程性质、目的与任务

本课程是作为程序设计第一语言为非计算机专业开设的计算机基础课程。通过该课程的学习使学生具备扎实的面向过程的程序设计的能力，为在以后的学习或工作中，能够使用 C/C++ 语言编程解决各自专业领域的计算机应用问题打下一个良好的基础。

二、教学目标

- 程序设计的基本概念与基本方法
- 编程解题的思路与典型方法
- 数学模型简介
- 算法及算法步骤
- 程序结构与相应语句
- 编码与上机调试

三、教学重点

- 在C/C++语言的环境下，学会如何**针对问题进行分析**，得出数学模型，理出算法并**编程实现**。
- **强化实践**：程序设计是高强度的脑力劳动，**不是听会的、也不是看会的，而是练会的**。这可能是与以往的教学安排最大的不同之处。
- 重点放在**思路、算法、编程构思和程序实现**上。语句只是表达工具，要求堂上积极思考，尽量当堂学懂，重在**训练分析问题和解决问题的能力**。

目录

1.1 程序和程序语言

1.2 C语言和C++语言简介

1.3 C++ 程序快速入门

1.4 集成开发环境 Dev-C++ 使用简介

1.1 程序和程序语言

对“程序”的直观理解：

- “程序 (program)” 一词来自生活，通常指完成某些事务的一种既定方式和过程。
- 按顺序实施这些步骤，即完成了该项事务。

例1：早晨生活

- 1、起床
- 2、刷牙
- 3、洗脸
- 4、吃饭
- 5、早自习

例2：到图书馆借参考书（更复杂的程序）

- 1、进入图书馆；
- 2、查书目；
- 3、填写索书单；
- 4、交图书馆工作人员取书；
- 5、如果书已借完，有两种选择：
 - 5.1 回到2（查找其他参考书的书目）；
 - 5.2 放弃借书，离开图书馆；
- 6、（有书）办理借书手续；
- 7、离开图书馆。

- “程序”的一些直观特征：
 - ◆ 按部就班地进行；
 - ◆ 开始与结束；
 - ◆ 完成某项具体任务；
 - ◆ 需要用某种记法形式描述（计算机程序需要用某种精确定义的形式描述）；
 - ◆ 是在一些基本动作的基础上描述的；
 - ◆ 不同的描述粒度（细节程度）；
 - ◆
- 把程序和编写程序的工作（programming）作为一件重要事情进行系统研究，主要是在计算机领域里。

计算机基本原理

- 现在的“通用电子数字计算机” (General - Purpose Electronic Digital Computer) 是用超大规模集成电路和其他元器件构造起来的一种复杂电子设备。

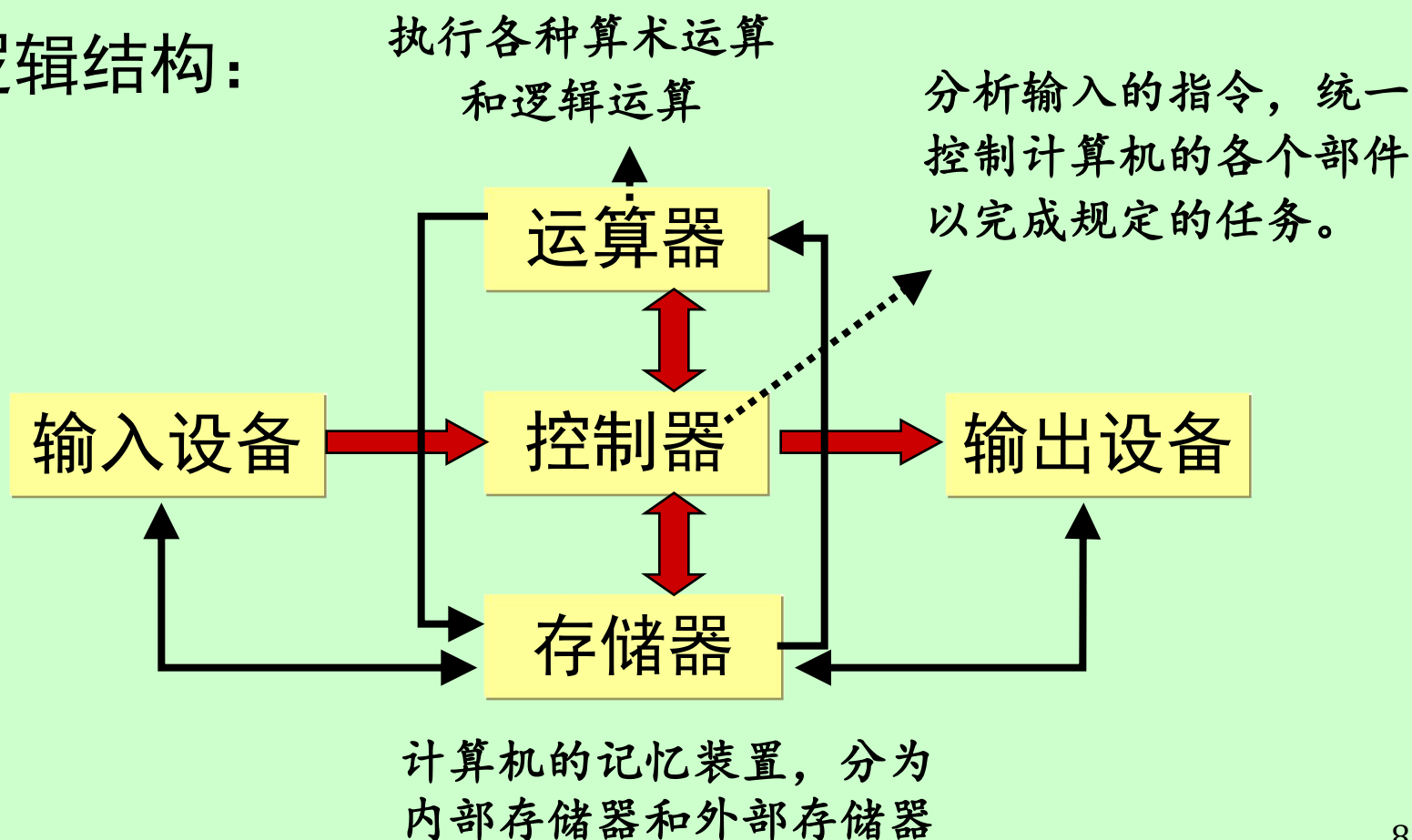
一个完整的计算机系统包括**硬件系统(Hardware)**和**软件系统(Software)**两大部分，依靠硬件和软件的协同工作来完成各种计算任务。

↓
依赖于计算机硬件的
程序及其相关数据

↓
计算机系统中看得见的
各种物理上的部件

计算机硬件系统采用冯·诺依曼体系结构，由**运算器**、**控制器**、**存储器**、**输入设备**和**输出设备**五个基本部分组成。它们通过**总线**连接。

逻辑结构：

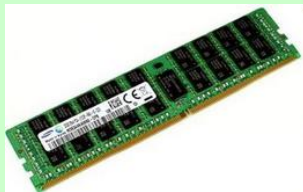


计算机物理结构:



输入设备

键盘、鼠标等、
光电输入



主板

中央处理单元
(CPU)

运算器

控制器

内部存储器

外部存储器

主机



输出设备

(显示器、打印机)



计算机能够自动地完成各种数值运算和复杂的信息处理过程的基础是**存储程序**和**程序控制**原理。



当需要计算机解决某个问题，就必须先把解决这个问题方法分解成一系列计算机所能完成的简单操作，并以指令（对计算机进行程序控制的最小单位）的形式通知计算机。这些**完成特定功能的指令序列**，就称为**程序**。

计算机程序的定义：**一组计算机能识别和执行的指令**。

计算机的基本工作原理简要描述为：

程序和被处理的数据由**输入设备**或外存装入**内存**；CPU在**控制器**控制下从内存中取出程序的指令，根据需要到指定地址取出所需数据，**运算器**执行指令要求的操作；运算结果存入内存，根据需要而通过**输出设备**输出。整个过程都在控制器的控制下进行。

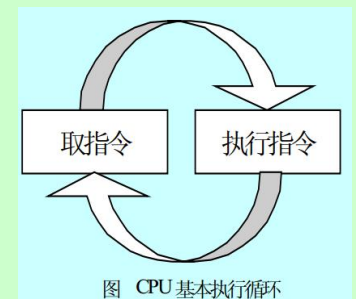
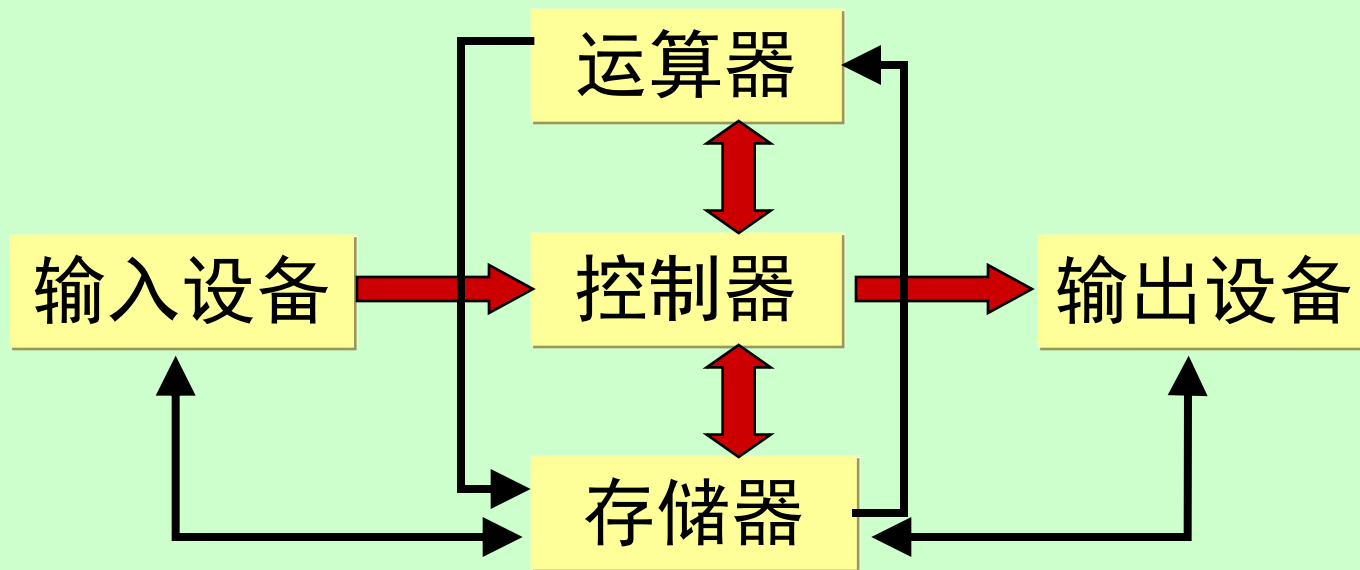


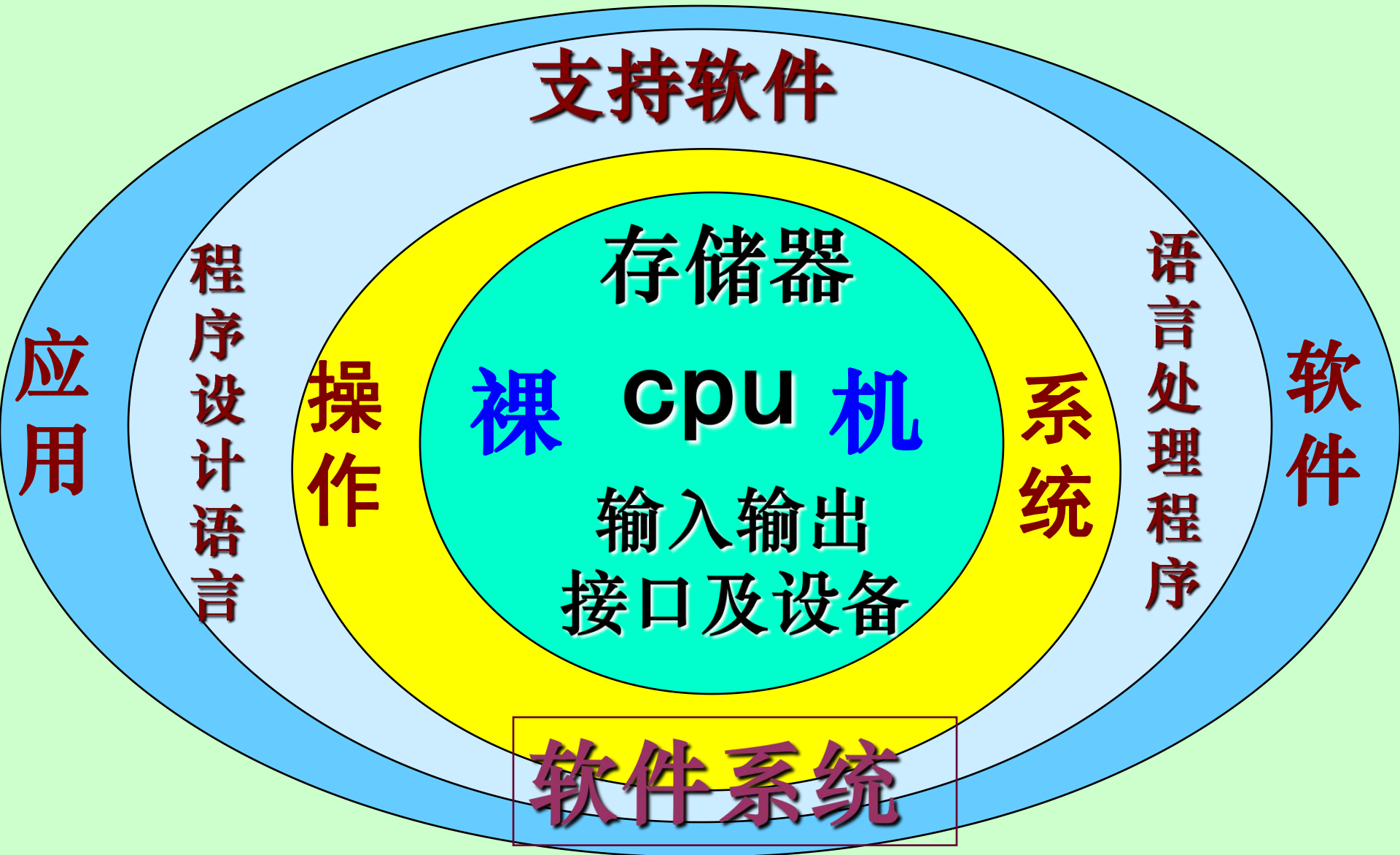
图 CPU 基本执行循环

软件系统

软件：依赖于计算机硬件的程序及其相关数据。

- 系统软件：计算机系统中供给用户使用的操作系统环境和控制计算机系统按照操作系统要求运行的软件。
 - 操作系统(Dos, Windows, Linux, ...)
 - 语言处理程序
 - 诊断程序
 - 数据库系统
 - 网络管理软件
- 应用软件：计算机系统支持下的所有面对实际问题 and 具体用户群的应用程序的总合。
 - 数据处理软件(如Matlab)
 - 文字处理软件(WPS、WORD)
 - 表格处理软件(如EXCEL)
 - 计算机辅助工程应用
 - 实时处理软件
 - 行业软件

计算机系统组成图示



- 计算机严格按照人们的指令进行工作。程序是人与计算机交流信息的最基本方式。人通过程序指挥计算机的活动。
- 编制计算机程序的工作称为**程序设计或编程(Programming)**，其产品就是**程序(Program)**。由于计算机的本质特征，从它诞生之初就有了程序
设计工作。
- 要用计算机处理问题，写程序时必须精确描述所需的全部细节，不能有一点含糊之处。
- 写程序需要用**程序设计语言(Programming Language)**。这种语言的特点是计算机可以处理，可以按它的指挥完成工作。程序设计语言是人与计算机交流的最基本最重要的媒介。

编程语言发展历史：

机器语言 → 汇编语言 → 高级语言

- **硬件层面**上的程序是**机器指令的序列**。
- 程序执行：将程序存入内存，通知CPU第一条指令的地址。命令它“开始”！
- 一般情况下，CPU执行完一条指令后，自动取出下一条指令，并如此继续下去。
- 转跳指令明确指定下一条指令的位置，人可以基于转跳指令描述复杂的执行流程。
- 人命令计算机去执行一个程序，计算机就会一丝不苟地按这个程序的内容，一条一条指令执行，直至程序结束（指令执行到了最后，或者遇到明确的停止指令）。

机器语言和程序

机器语言是机器指令形成的语言；
形式为二进制编码，机器可直接执行。

00000001000000001000	数据装入寄存器0
00000001000100001010	数据装入寄存器1
00000101000000000001	寄存器0与1的数据乘
00000001000100001100	数据装入寄存器1
00000100000000000001	寄存器0与1的数据加
000000100000000001110	保存寄存器0里的数据

难写难读，人使用不便，程序开发效率极低。

机器语言 → 汇编语言 → 高级语言

汇编语言：

采用助记的符号形式，有利于人的阅读和使用。

汇编指令与机器指令一一对应

load 0 a	将单元 a 的数据装入寄存器 0
load 1 b	将单元 b 的数据装入寄存器 1
mult 0 1	寄存器 0 与 1 的数据乘
load 1 c	将单元 c 的数据装入寄存器 1
add 0 1	寄存器 0 与 1 的数据加
save 0 d	将寄存器 0 里的数据存入单元d

- 计算机无法直接执行汇编语言程序，执行前需要把汇编语言程序翻译为机器指令程序
- 最早时通过手工翻译为机器指令，后来人们开发出称为“**汇编系统**”的程序，让计算机去完成程序翻译工作。

汇编语言的特点：

- 每条指令的意义容易理解
- 程序粒度太小，细节太多
- 程序无结构，缺乏组织手段
- 写大程序仍然很困难

高级语言和程序

高级语言的特点：

- 具有类似文字的表现形式
- 用类似数学的表达式形式描述基本计算
- 用变量等概念取代低级的存储概念，使人摆脱各种繁琐低级的工作，例如存储的安排
- 提供高级操作流程控制手段和程序组织手段

在 C 语言里写前面同样的程序：

```
d = a * b + c;
```

机器语言 → 汇编语言 → 高级语言

- 计算机无法直接执行高级语言写出的程序。
- **编译型语言**：做一个**编译系统**，完成高级语言程序到机器语言可执行程序的翻译加工：

编程 → 编译加工 → 执行

- **解释型语言**：做一个**解释器**，对程序边解释边执行

编程 → 解释并执行

- 使用高级语言编程：
 - ◆ 编程工作的效率大大提高
 - ◆ 人更容易思考和把握复杂程序的意义
 - ◆ 更多人愿意投身于这种工作，使编程发展成为一种职业和谋生方式
- 术语“程序设计语言”已专指“高级语言”。

高级语言的简单历史

- 1954年到1957年：高级语言 **FORTRAN** 诞生
- 至1960年代中，开发了Algol 60，COBOL，**BASIC** 等。还有函数式语言LISP 等
- 1970年代开始：Pascal 和 **C** 语言逐渐分别为教学科研和软件开发用的主要语言
- 1980年代：逻辑程序语言Prolog，面向对象语言Smalltalk 和后来的C++。另有ML等
- 1995年左右的 **Java**
- 还有很多脚本语言，如**Perl**、**Python**、**Ruby**。

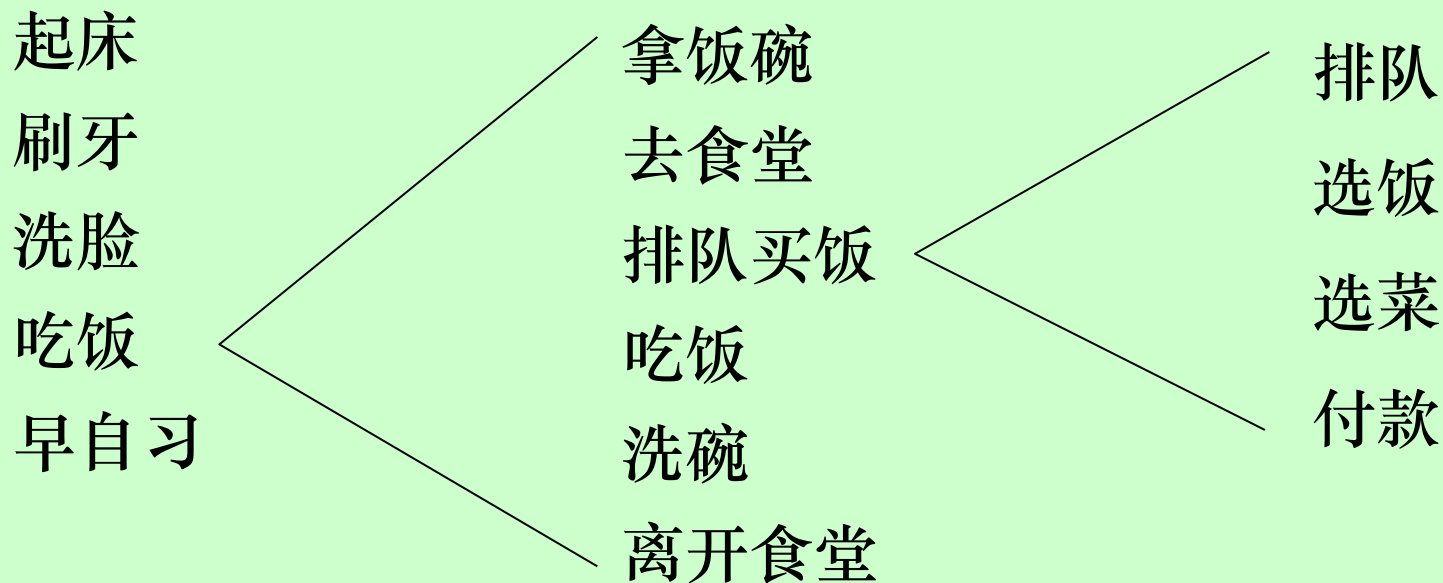
本课程的要点

- 用计算机解决问题的过程和基本方法
- 程序设计的基本方法
- C/C++语言的规定和性质
- 怎样写好C/C++语言程序
- 程序设计过程和一些基本技术

程序设计过程

- 编写程序不应该从第一个细节开始
- 逐步分解，直至分解到程序语言提供的功能。

“程序”分解实例（早起活动）：



编程的工作方式：

- 从问题出发，从高层开始设计程序；
- 逐步分解程序功能，直至可以用程序语言实现。

需要学习和理解：

- 程序语言所提供的基本功能；
- 各种语言功能的形式和意义；
- 所用编程工具（C/C++编程环境）和使用技术；
- 程序设计的典型技术。

写好程序：1) 模仿好的范例，2) 实践。

注意前人经验，包括程序书写形式和许多具体写法等

注意养成写程序的良好习惯。书中许多地方提出了相应的建议

写出程序和**写好程序**之间有很大距离

只有**写好**小程序，才能**写出**大些的程序

1.2 C语言和C++语言简介

简单历史

- C语言1973年由贝尔实验室的 Dennis Ritchie 设计，目标是书写操作系统和其他系统程序。
- C语言最早用于写 UNIX 系统。70年代成为 UNIX 的标准开发语言，随 UNIX 流行而被广泛接受。
- 80年代被搬到各种机器的许多操作系统上，逐渐成为一种开发系统程序和复杂软件的通用语言。
- 后来成为使用最广泛的系统开发语言。被用于开发各种程序，从简单应用到极复杂的大型软件。
- 各种计算机都有可用的 C 语言系统。

- C 语言比较小，入门容易，很快就可以开始编程
- 有丰富的程序机制、数据机制、函数定义机制，能满足复杂程序的需要。许多常用功能通过库实现
- 提供接近硬件的低级操作，广泛用于开发效率要求高的程序。被用于代替汇编语言开发底层软件
- 提供了一些支持大规模复杂软件开发的机制
- C 语言的工作得到世界计算机界的广泛赞许。对计算机工业和应用发展起了重要推动作用
- 许多新语言从 C 汲取营养。如C++，Java等
- C语言设计者获得计算机领域最高奖——图灵奖

标准化

- 应用发展要求 C 成为更安全可靠、不依赖具体机器或操作系统的标准语言。
- ANSI在80年代开始标准化工作，1988年颁布ANSI C标准，后被ISO和各国接受，也采纳为中国国家标准。
- 新标准C99已经通过。

按ANSI C标准写程序

- C 的原设计注重灵活性，允许许多不安全的编程方式，正确性靠编程者。用户群扩大后缺点凸现，复杂程序常有隐藏错误。标准化也是为修正C的缺陷。
- ANSI C基本容许原程序形式。我们应该采用标准所提倡的形式。这样做也更容易完成编程工作。

1.2 C/C++语言的发展简史和特点

1. C语言的诞生与发展

(1) 在C语言诞生以前，系统软件主要是用汇编语言编写的。其可读性和可移植性都很差；但一般的高级语言又难以实现对计算机硬件的直接操作。

(2) C语言：Bell实验室的 **D.M.Ritchie** 于1972年为了编写UNIX设计的。后来又被多次改进，并出现了多种版本。

几个重要的标准：

- K&R 标准：1978年，《The C Programming Language》
B. W. **K**ernighan 和 D. M. **R**itchie 合著
- 1980年代初，美国国家标准化协会制定了 **ANSI C** 标准
- 1999年再次做了修订，称为 **C99**。

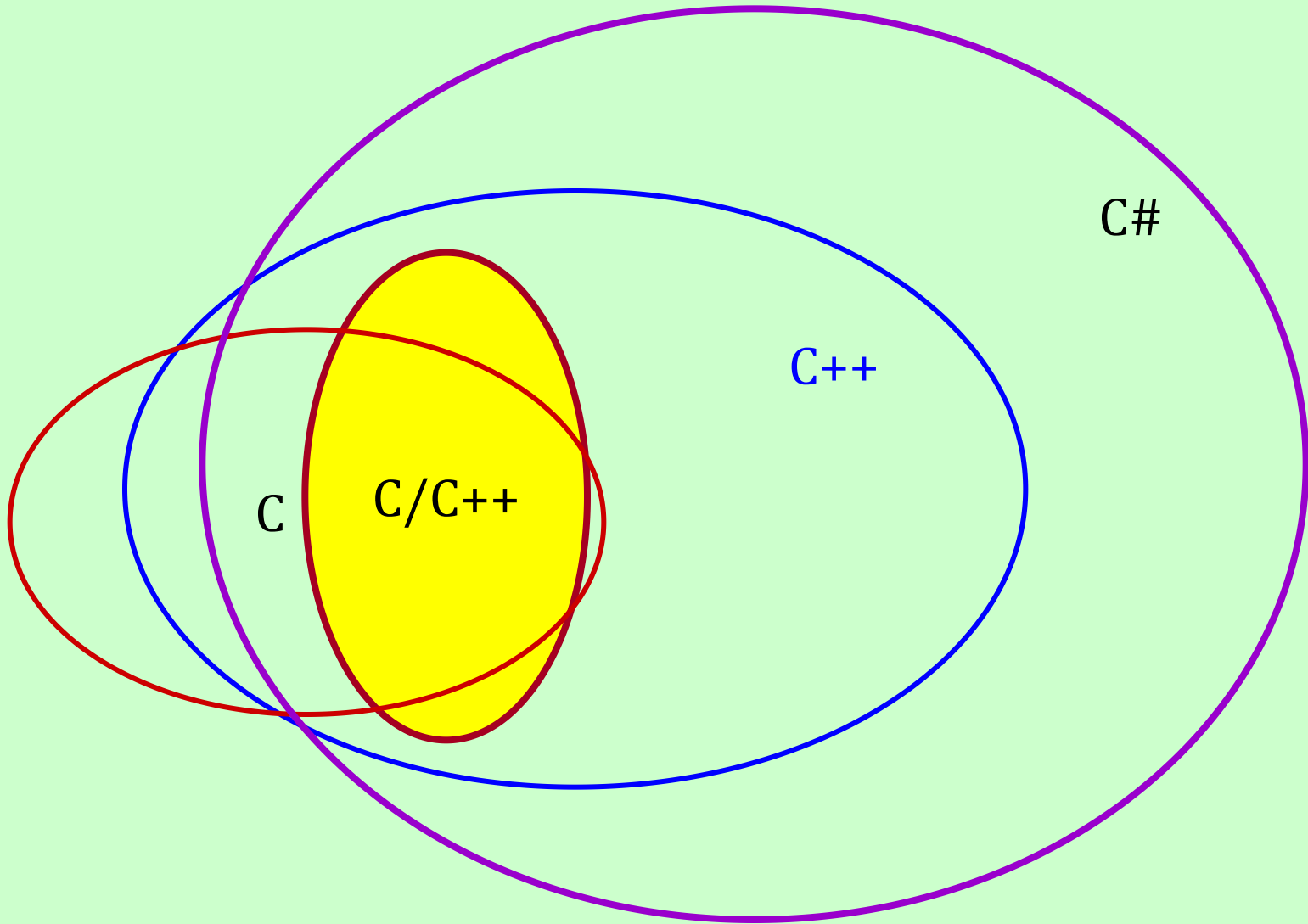
- 在 C 的基础上，1983 年又由贝尔实验室推出了 C++。C++ 包含了整个 C，进一步扩充和完善了 C 语言，添加了对面向对象编程的完全支持。
- 通常，初学者主要学习和使用 C 语言的功能，偶尔也会学习和使用少量的 C++ 语言的功能。
- C/C++：C 语言和 C++ 语言共有的功能和特性。
- C# 是微软公司发布的一种由 C 和 C++ 衍生出来的面向对象的编程语言。它在继承 C 和 C++ 强大功能的同时去掉了一些它们的复杂特性。是兼顾系统开发和网络应用开发的语言。

C/C++ 语言的优点

- 语言简洁、紧凑，使用方便、灵活，程序书写形式自由；
- 把程序中需要的许多功能放在程序库（称为标准函数库）；
- 结构化的体系结构。层次清晰，便于按模块化方式组织程序，易于调试和维护；
- 非常强的处理能力，运算符丰富，代码效率高；
-

C/C++ 语言的缺点

- 太灵活，不易掌握，容易出错；
- 运算符优先级太多；
- 类型转换限制少，检验较弱，不够安全；



● 小结

1.3 C++ 程序快速入门

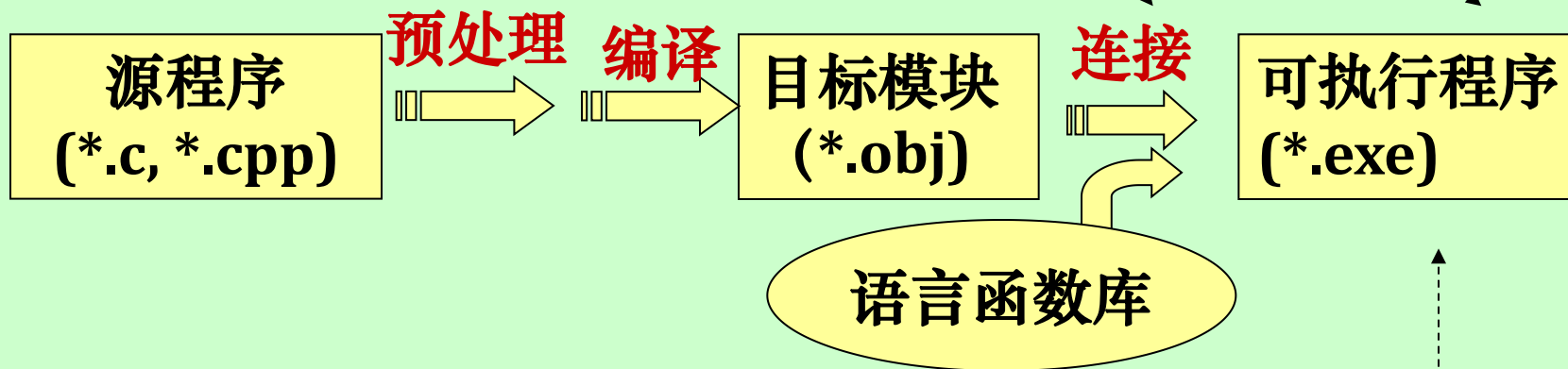
- C /C++ 是高级程序设计语言，其开发过程要经过四个步骤：
 - 1、**编辑(Edit)**：编程人员把按照 C/C++ 语言语法规则编写的程序代码通过文本编辑器输入计算机并保存。→“**源程序**”。
 - 2、**编译(compile)**：将编辑好的 C/C++ 源程序通过编译器转换为目标文件，即生成该源文件的**目标代码**。
 - 3、**连接(link)**：将用户程序生成的目标代码文件和系统提供的库文件中连接在一起，生成**可执行文件**；
通常把**编译**和**连接**合起来称为**构建 (build)** 或**生成**。
 - 4、**运行(run)**：运行生成的可执行文件，在屏幕上显示运行结果。用户可以根据运行结果来判断程序是否工作正常。

分析思考

编程人员

编辑

执行



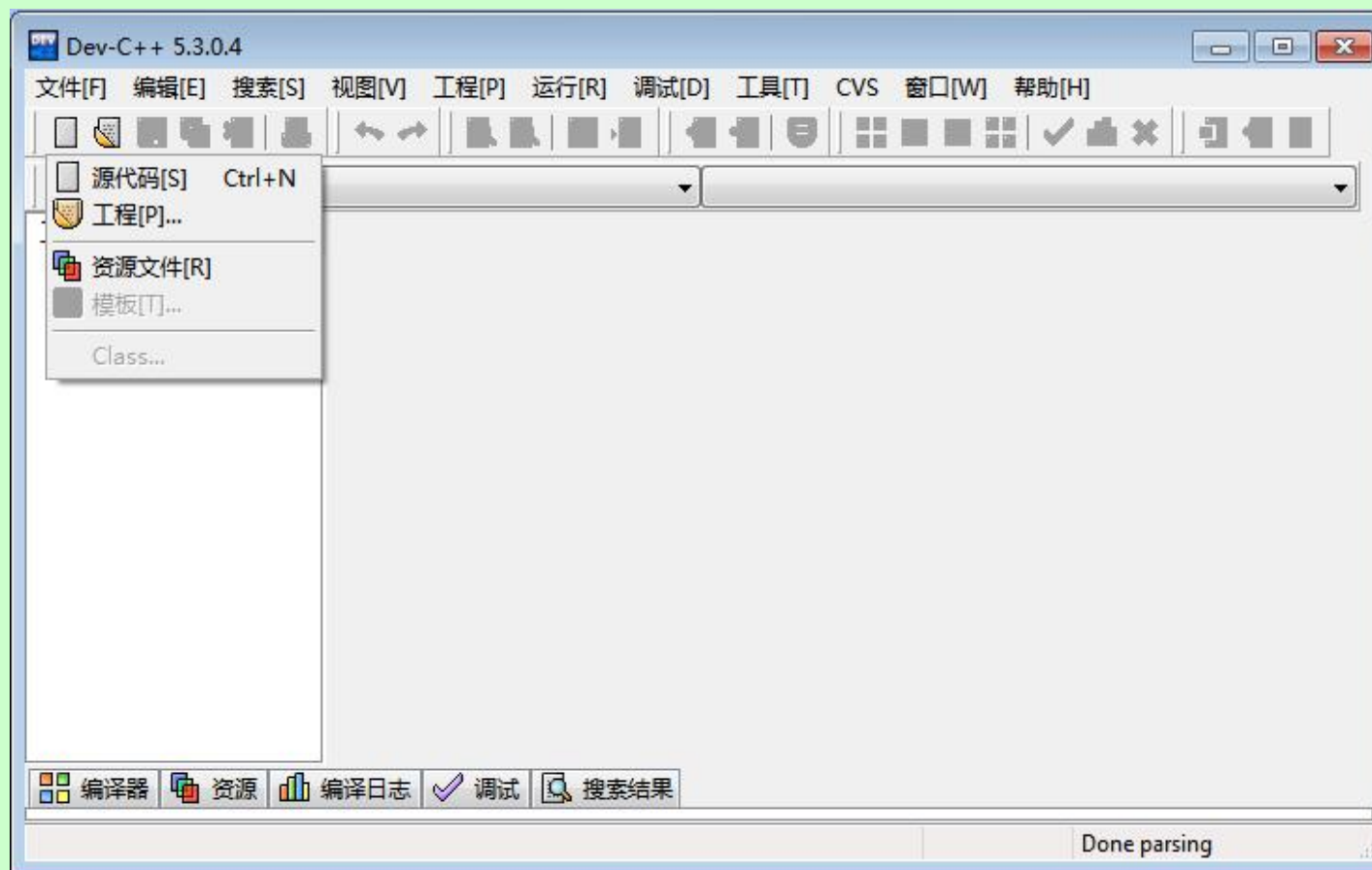
可执行程序是所有软件的核心。在用户的操作下，它们能忠实地按照编程人员的设计，执行预定的功能。

集成开发环境

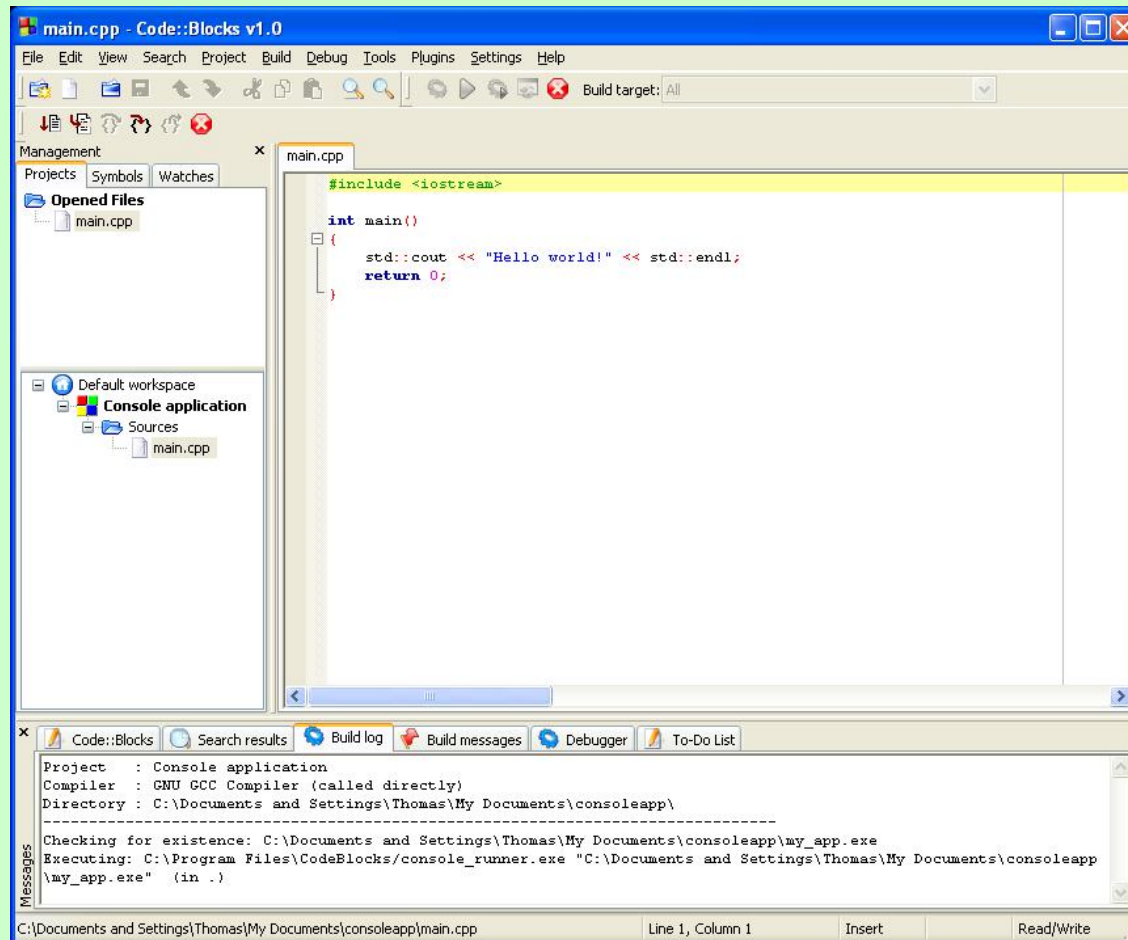
- 编程时需要程序进行编辑、编译、连接、执行、调试等各项操作。
- 这些操作可以使用多个独立的软件进行，但更常见的是使用集成开发环境（IDE：Integrated Development Enviroment）——集成了文本编辑、程序编译链接（构建）、程序执行、程序调试的软件。
- 目前常用的IDE有：
 - ◆ Microsoft公司出品的Visual Studio（有各种版本，还分为商业版和学习版）。
 - ◆ 包含免费编译器GCC的 [Dev-C++](#)、[Code::Blocks](#)。
 - ◆ 在安卓系统的手机上，可以安装使用 C4droid 或其它

- **Visual Studio** 是微软公司推出的一个基于Windows操作系统的功能强大的可视化软件集成开发环境。
- 它由许多组件组成：
 - ◆ 可视化工作室 “Visual Studio”（编辑器、调试器等）
 - ◆ 微软公司开发的 C++、C#、.net 编译器
 - ◆ 微软公司开发的一套VC Windows控件
 - ◆ 程序向导AppWizard、类向导Class Wizard等辅助开发工具。
- 国内仍然普遍使用 1999 年发行的 6.0 版本(Visual C++ 6.0)。它比较简单，适合于初学者使用。
- 当代版本称为 “**Visual Studio**”，功能非常强大而复杂。给学生使用的免费版本称为 Visual Studio Community（旧名称：Visual Studio Express）。

- **Dev-C++ (Dev-Cpp)** 是一个 Windows 环境下的 C/C++ 集成开发环境(IDE)。它集成了编辑器、源代码格式化工具 AStyle、编译器 GCC、调试器 GDB 等多种工具，适合于小型 C/C++ 程序开发。



- **Code::blocks** 是一款开源的跨平台集成开发环境。通过配置不同的编译器，可以支持包括 C/C++ 在内的各种编程语言。它提供了许多工程模板并支持各种插件。



该选用哪个集成开发环境？

- 微软 Visual Studio 是基于“Project”（工程，项目）概念的，一个工程可以包含许多文件，适合于编辑由多个文件构成的工程。在使用时，VC 会给每个工程建立工程文件和工作区文件，从而产生一些额外的文件。
- 初学者/科研工作者通常都只生成在命令提示符环境下运行的文件，因此用不上 VC 中的控件。只有在需要生成图形界面的程序时才需要使用 VC 中的控件。
- Dev-C++ 和 Code::Blocks 比较适合于只编辑几个文件的初学者。使用时不会产生额外的文件。
- 本课程的教学主要使用 Dev-C++。

- 有初学者说：我学的是dev-c++，你学的是VC，咱们学得不一樣.....
- 这里实际上有一个重要的误解，就是混淆了 C 语言和 C 语言的程序开发系统（针对该语言的程序开发环境）。
- 这两者之间到底有那些不同？我们在学习程序设计时应该如何认识这个问题呢？
 - (1) C/C++语言是抽象的，而语言环境是具体的。
 - (2) C/C++语言标准提供的是语言的基本定义，而各个程序设计环境可能有许多自己的扩充。
 - (3) C/C++ 语言本身反映的是一般性的程序设计实践，而具体程序设计环境又在此之上填加了许多与具体系统有关的东西、基本程序设计之外的东西。

- 需要提醒的是，我们虽然在学习中使用特定的一个集成开发环境(IDE)，但读者**不要把自己局限在这个 IDE 下**。
- 应该了解不同的 C/C++编译系统的特点和使用方法，在需要时能将自己的程序方便地移植到不同的平台上。

● 小结:

1.3 C++ 程序快速入门

【例1-1】 编程在屏幕上显示"hello, world!"

```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

- 编辑如上文件。注意：C/C++是大小写敏感的语言（大小写字母是不同的），编辑时请注意中英文字符
- 保存文件，扩展名为“.cpp”。

```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

(1) 注释

- 以 `//` 开始的单行注释
- 以 `/*` 开始、以 `*/` 结束的块式注释。
- 注释只是给人看的，对编译和运行不起作用。所以可以用汉字或英文字符表示，可以出现在一行中的最右侧，也可以单独成为一行。

```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

(2) #include <iostream>

- 告诉编译器(后文介绍)把一个叫 `iostream` 的系统文件包含进来，这样就可以调用 `cout <<` 了。如果缺少这句，编译器就不认识 `cout <<` 。

```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

(3) using namespace std;

- 使用命名空间 std

```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

程序分析:

- (4) `main()`: 主函数，是程序的基本部分；
- 每一个 C 程序必须有、且只能有一个 `main` 函数；
- 无论 `main` 函数放在文件中什么位置（开头、中间或最后），程序总是从 `main()` 函数开始运行。


```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

- main() 函数的主体只包含一条语句：调用 `cout <<` 进行输出。
- 语句功能：在输出窗口按原样显示 “ ” 中的字符序列： `hello, world!`

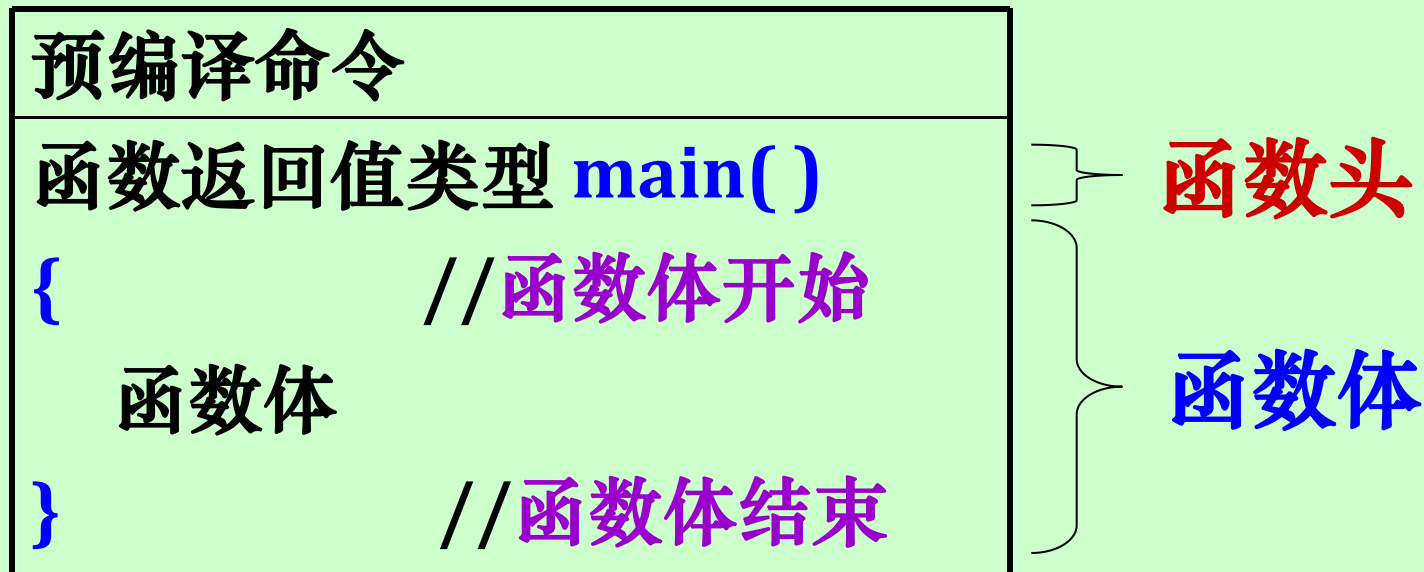
```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

(5) return 0;

- 程序运行完毕时，通常要返回给系统一个值，把自己的运行情况告诉系统。通常以 0 表示工作正常。

```
/* 我的第一个程序：在屏幕上输出字符串 */  
#include <iostream>  
using namespace std;  
int main () {  
    cout << "Hello, world!" << endl; //屏幕输出  
    return 0;  
}
```

- 由上例可知，C /C++ 程序的一般结构如下：



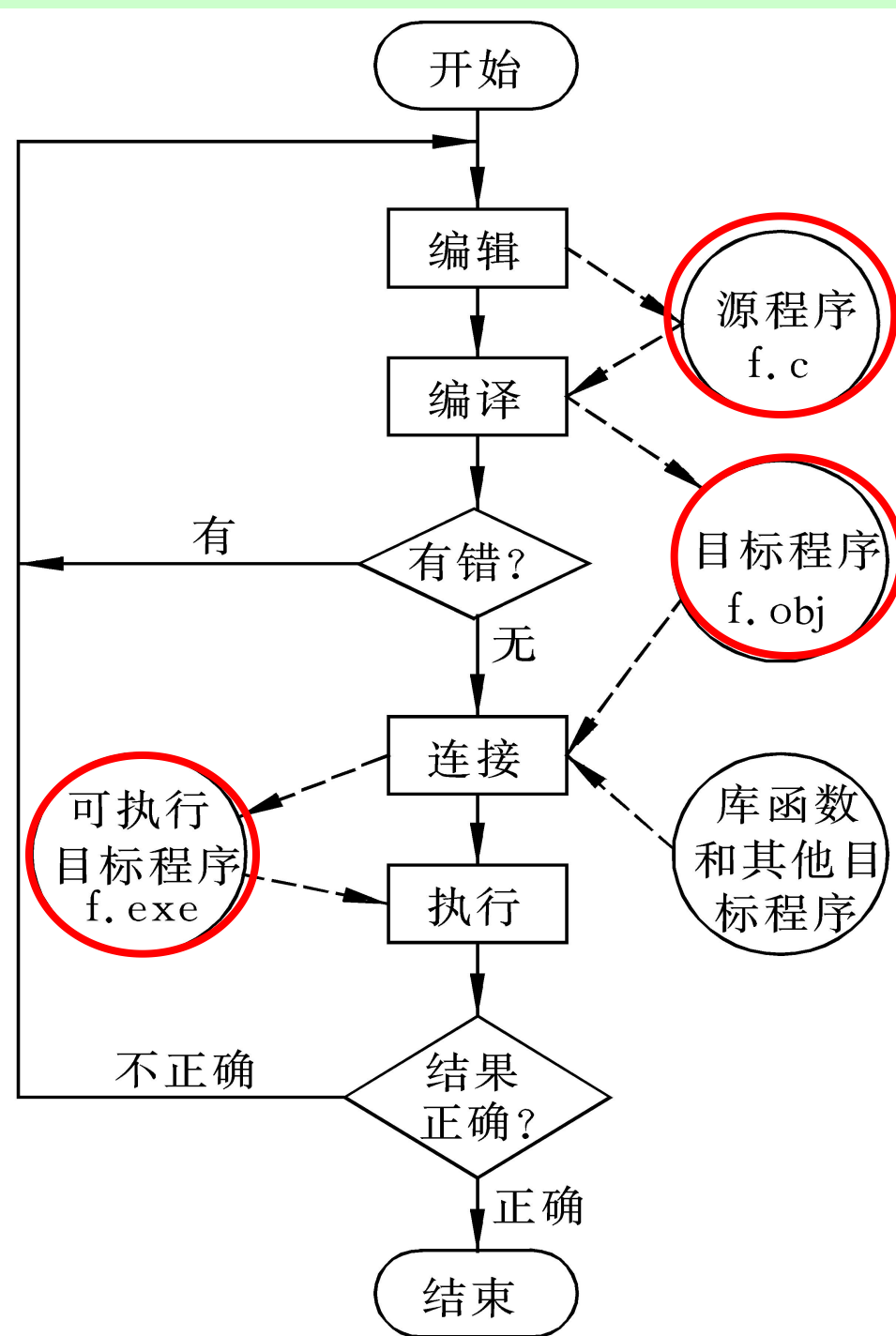
语句与复合语句：

- 程序中对计算机的操作是由函数中的 **C 语句** 完成的。
- **语句** 是程序的基本单位，每条语句以分号为结束符，分号是语句的一部分。
- C 程序书写格式自由，一行内可以写几个语句，也可以把一个语句分开写在多行上。
- 用一对花括号把 0 个或多个语句括起来，就构成了 **复合语句**（复合结构）。

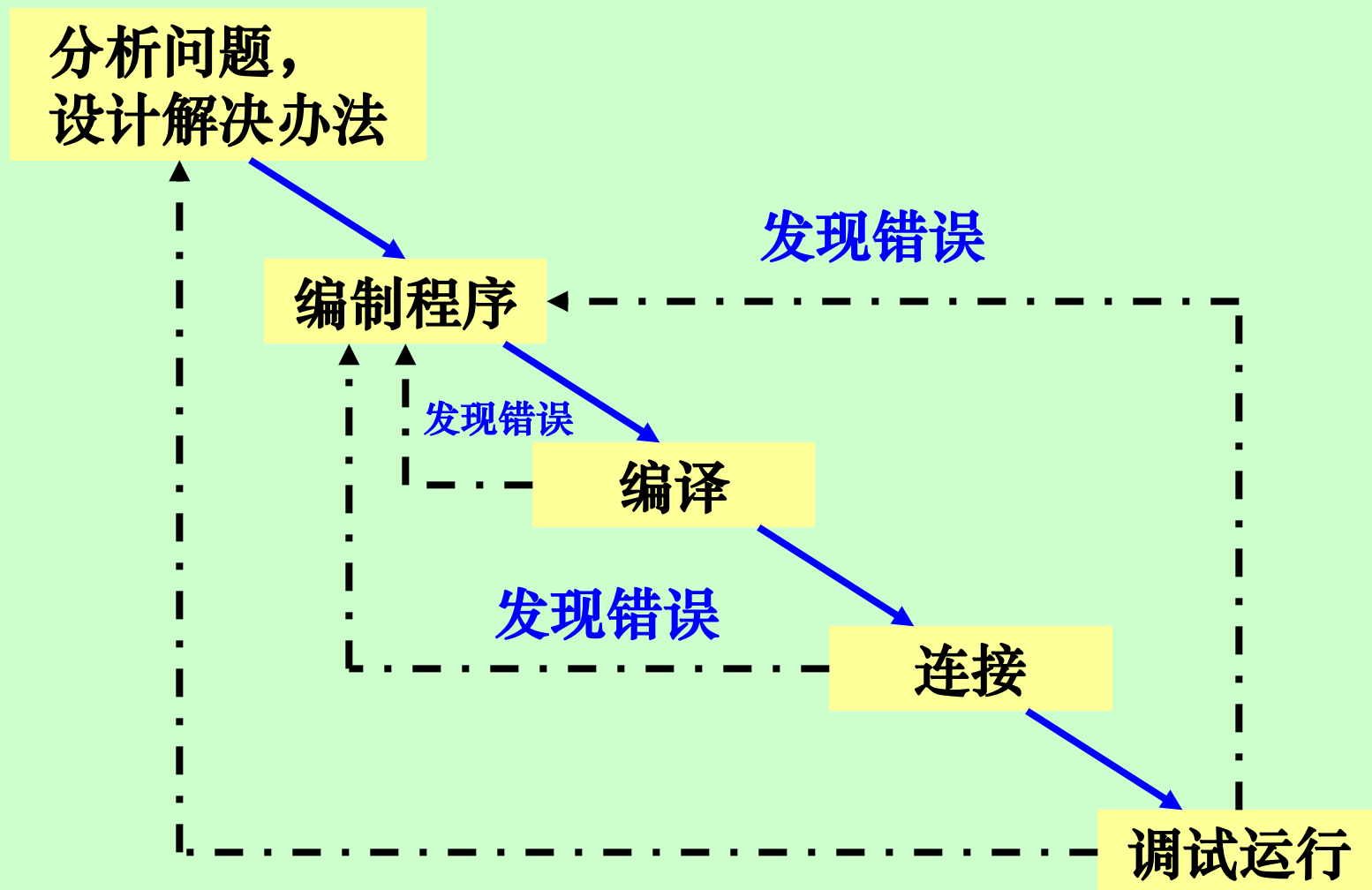
- C和C++ 都是“自由格式”语言。
- 由于程序可能很长，结构可能很复杂，因此必须采用良好格式写出，所用格式应很好体现程序的层次结构，反映程序中各个部分之间的关系。
- 人们普遍认可的程序格式是：（1）在程序里适当加入空行，分隔程序中处于同一层次的不同部分；（2）同层次的不同部分相互对齐排列，下一层内容适当缩进（在一行开始增加制表符或空格）并相互对齐，使程序结构更清晰；（3）在程序里加一些注释。前面程序例子的书写形式符合这些看法。
- 读者在开始学习程序设计时就应养成注意程序格式的习惯。

程序开发过程：

程序从**编辑**到最终顺利**执行**，通常需要经过反复修改，排除错误，这个过程称为**程序调试**。



● 程序开发过程（另一种图示）



- 有些初学者对编程工作有一种常见的误解，以为编程序就是“写程序、编译、运行、完事”。这种理解是错误的。
- 应该强调：
 - (1) 程序开发工作的第一步是分析问题并设计解决方案，这是一项重要的脑力劳动。只有经过充分思考并设想出了合理的解决办法，才能动手开始编写程序。
 - (2) 程序常常可能含有一些明显的或隐藏的误差，因此需要进行程序除错(debug)和调试。此外，完成了的程序，也常常需要进一步完善或修改扩充。这些也是重要的编程工作。

五、程序除错

- 程序中的错误，其实都是**编程序的人自己犯的错误**，并没有其它客观原因。所谓程序除错，也就是找到并清除自己开发程序的过程中所犯的错误，或说是消除自己写在程序里的错误。
- 程序中的错误可以大致分为两类：
 - ◆ 语法错误
 - ◆ 逻辑错误
- 需要熟练掌握程序开发系统的使用方法，更需要积极开动脑筋，认真观察、分析和思考。

1.4 集成开发环境 Dev-C++ 使用简介

- 一、源程序的编辑、保存、关闭和打开
- 二、源程序的加工和运行

老师演示第一次上机操作的过程。

- 下载并安装 Dev-C++ 软件
- 编辑并保存源程序
- 用AStyle工具格式化源程序
- 编译、运行
- 写心得体会
- 上传文件到FTP
- 自行保存文件

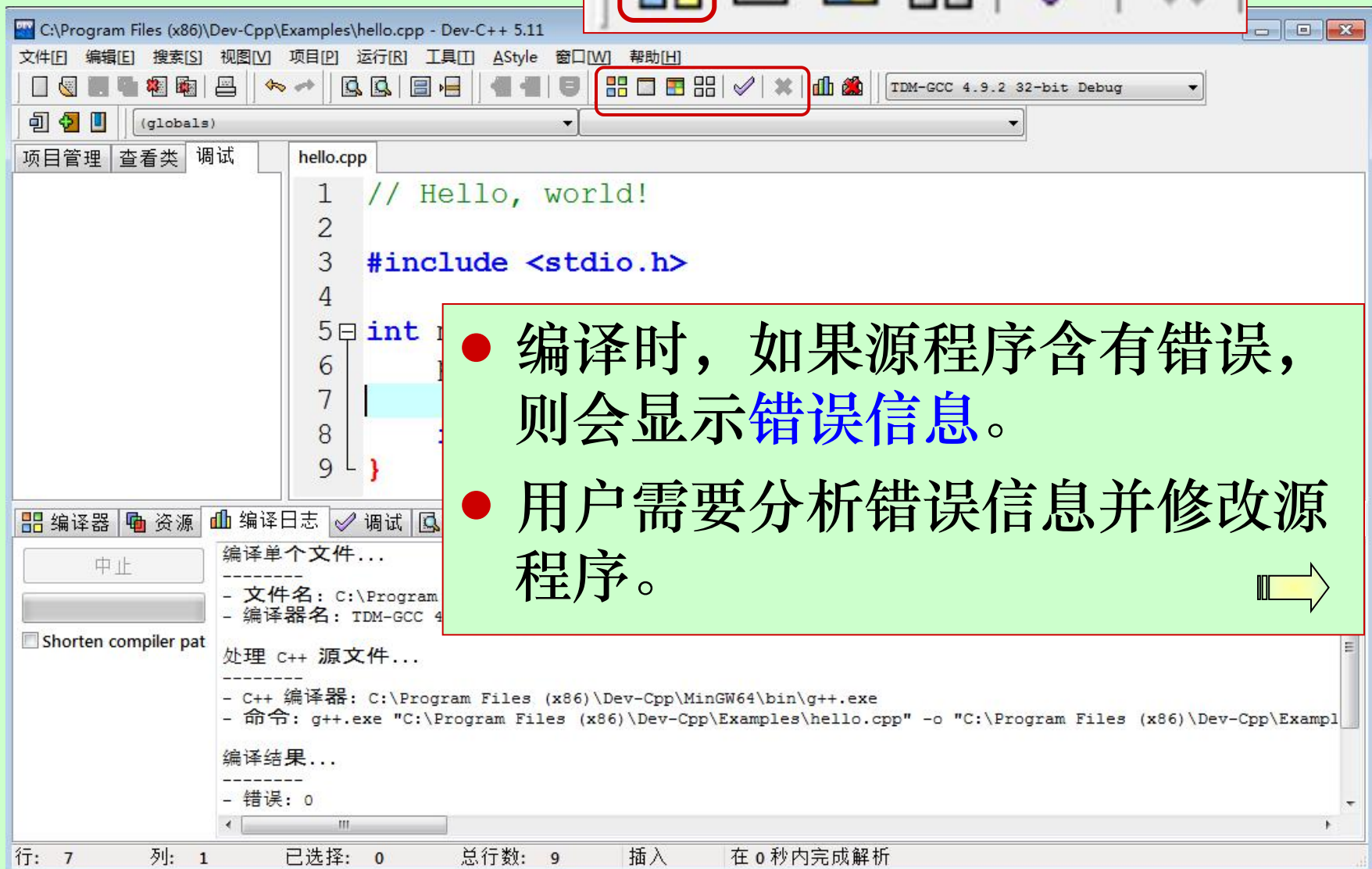
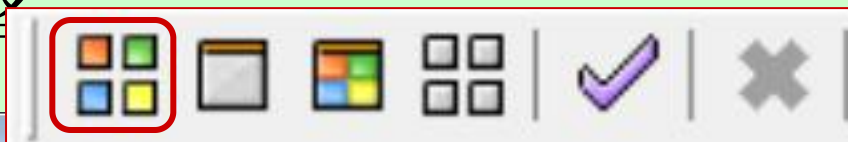
上机要求

- 开始学习时应该严格按照老师的要求去做。
- 每编写一个程序，都要在程序头部以注释的形式写上自己的学号、姓名和日期，最后要补充写上学习体会。
- 编写好的程序文件要保存起来，文件名有要规则，方便自己以后查看。（不能给文件命名为“未命名1”、“新程序1”、“Untitled-1”这样无意义的名称，文件夹也不能命名为“新建文件夹1”之类的）
- 程序文件要上传到机房的 FTP 服务器上供老师检查。
- 程序文件还要以某种合适的方式自行保存：可以保存到自己的优盘上，也可以发送到自己的邮箱中。
- 如果用优盘，请在优盘上写上个人信息，以便丢失时别人可以送还。

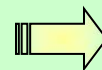
初学者在编辑源程序时需要注意的事项：

1. 在源程序中添加一些空行有利于查看；
2. 在源程序中应该严格地使用缩进：键入制表符(Tab) 或使用退格键(BackSpace)。
3. 虽然在 Dev-C++ 中可以使用 AStyle工具 格式化全文，但那是事后补救。正常情况下应该手工设置缩进，有助于理清编程逻辑。
4. 在编辑时要灵活切换中/英文输入法。

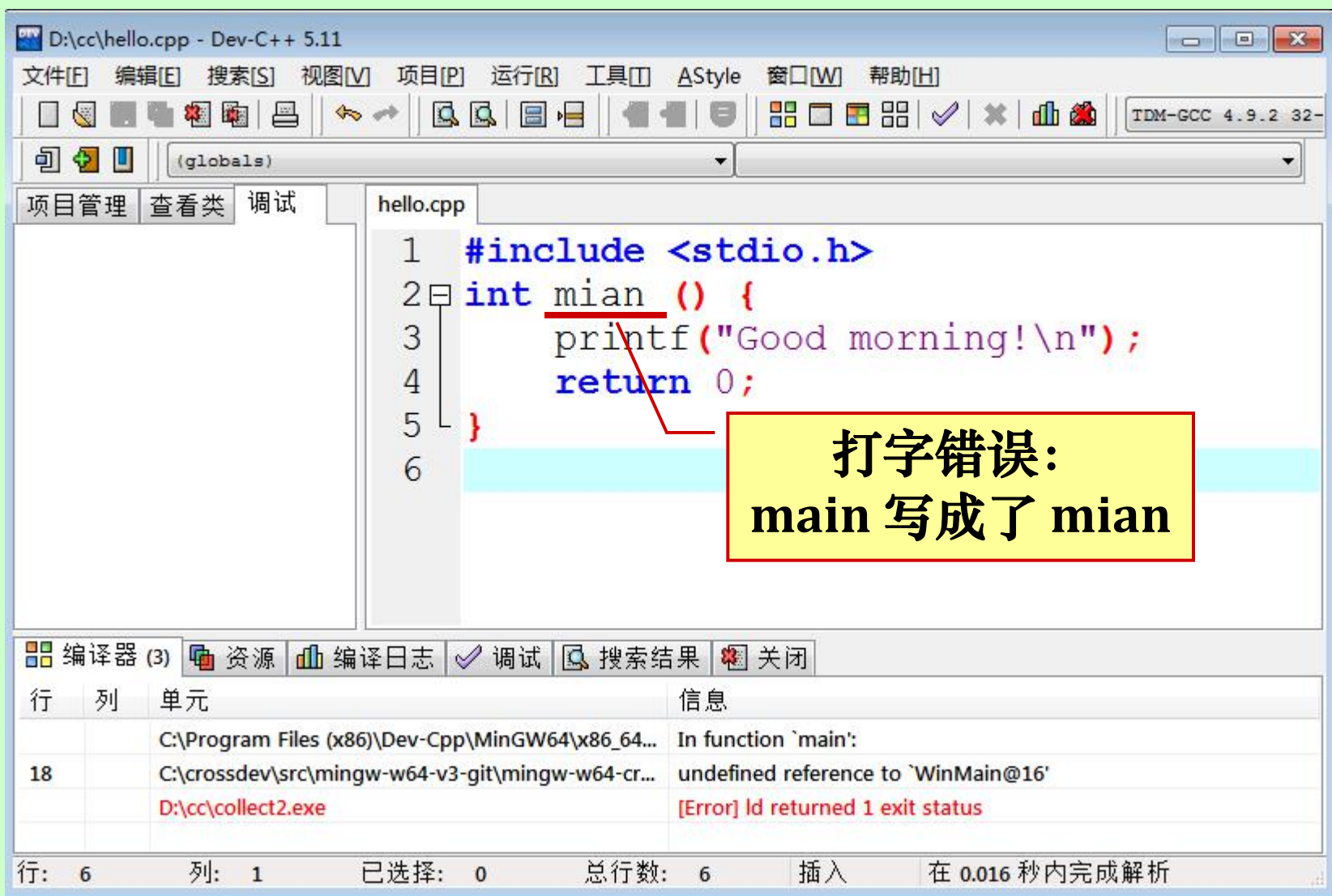
- 完成编辑之后，点击“**编译**”按钮就可以调用GCC 编译器进行编译



- 编译时，如果源程序含有错误，则会显示**错误信息**。
- 用户需要分析错误信息并修改源程序。



初学者容易编译出错的例子(1):



The screenshot shows the Dev-C++ 5.11 IDE with a C++ file named `hello.cpp`. The code contains a typo in the function name:

```
1 #include <stdio.h>
2 int mian () {
3     printf("Good morning!\n");
4     return 0;
5 }
6
```

A red arrow points from the yellow box to the word `mian` in the code.

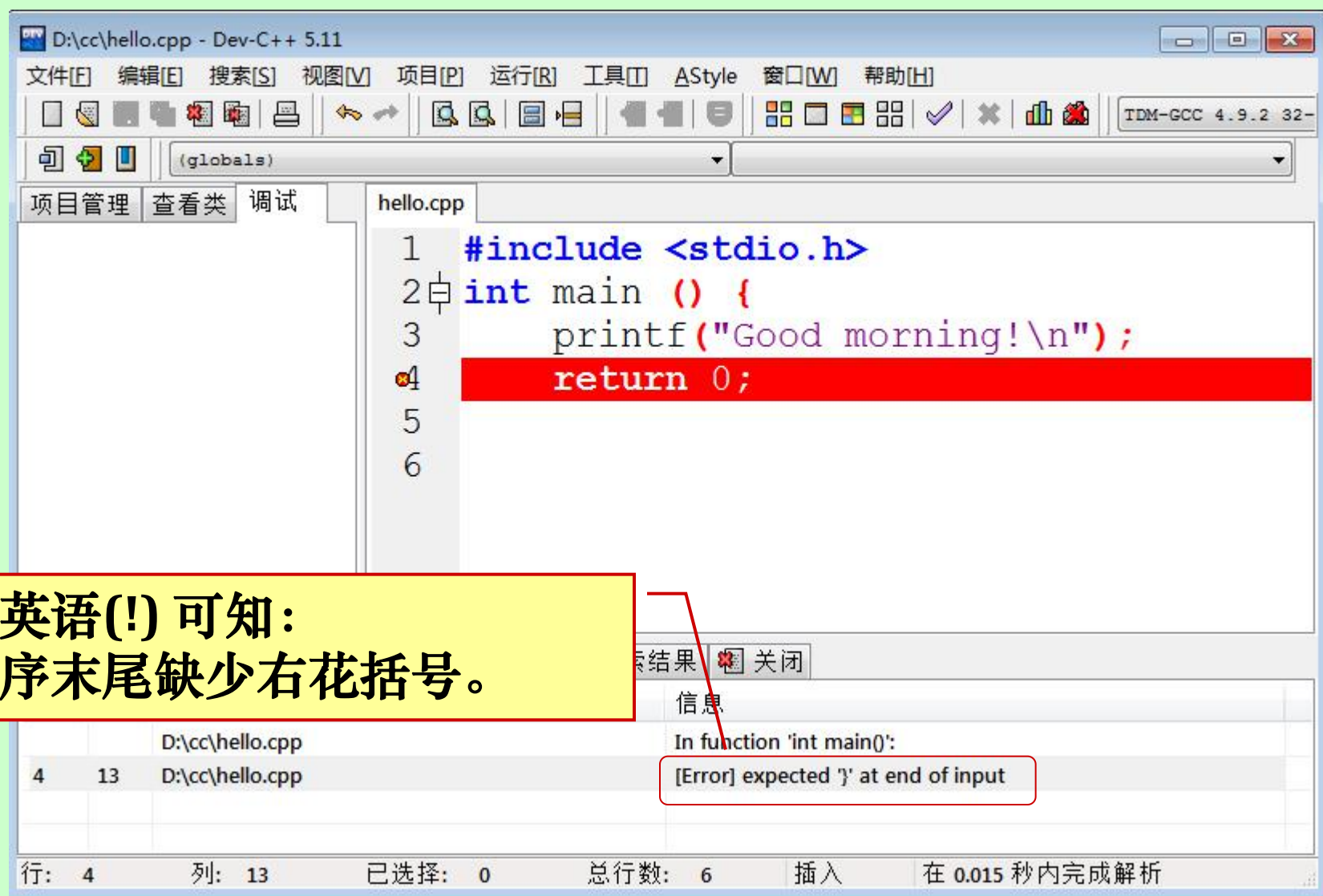
**打字错误:
main 写成了 mian**

The bottom panel shows the compiler output with the following error:

行	列	单元	信息
18		C:\crossdev\src\mingw-w64-v3-git\mingw-w64-cr...	In function `main': undefined reference to `WinMain@16'
		D:\cc\collect2.exe	[Error] ld returned 1 exit status

At the bottom, the status bar shows: 行: 6 列: 1 已选择: 0 总行数: 6 插入 在 0.016 秒内完成解析

初学者容易编译出错的例子(2):



初学者容易编译出错的例子(3):

虽然错误信息对应着第 4 行，但实际上是第 3 行末尾缺少分号。

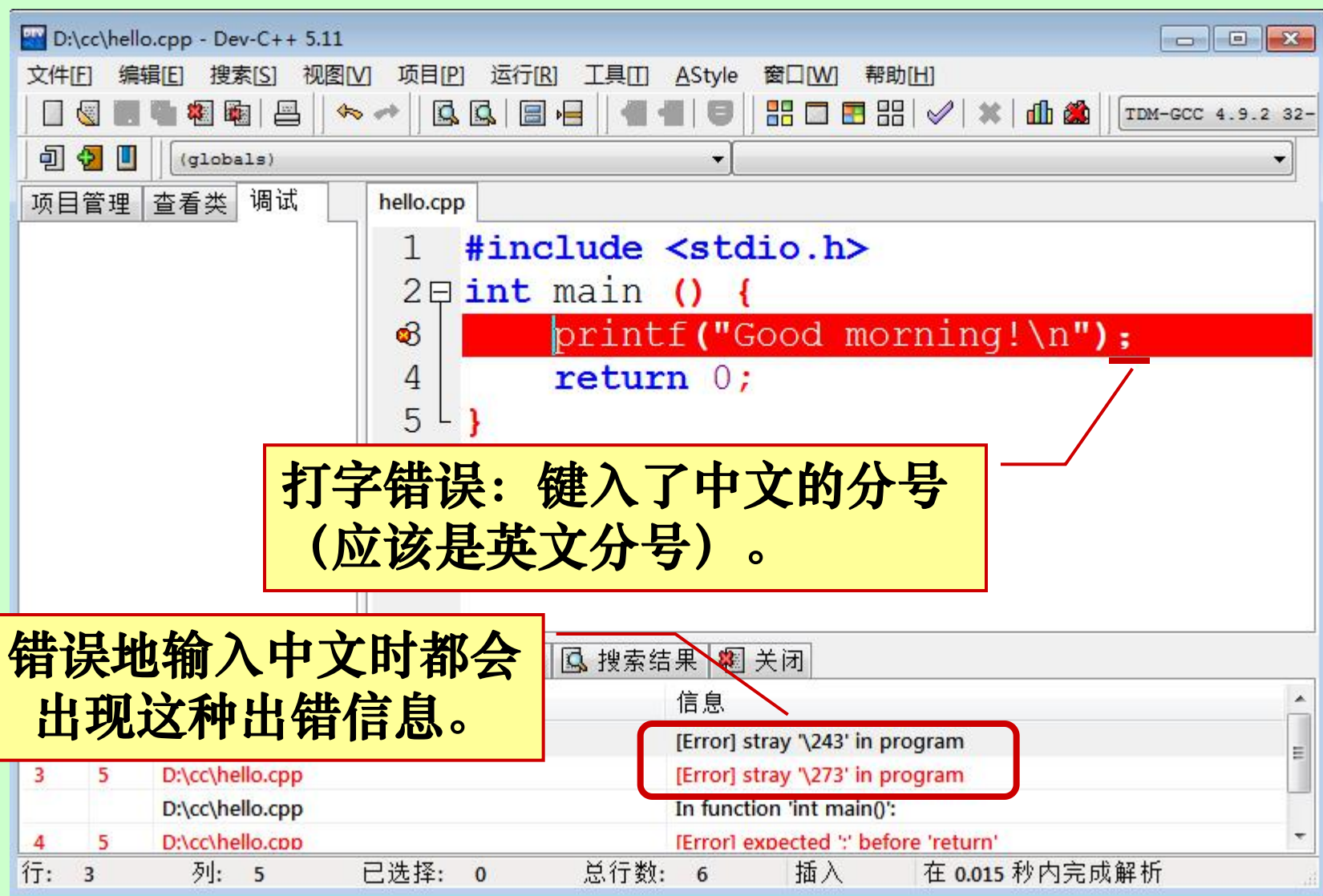
```
1  #include <stdio.h>
2  int main () {
3      printf("Good morning!\n")
4      return 0;
5  }
```

编译器 (2) 资源 编译日志 调试 搜索结果 关闭

行	列	单元	信息
		D:\cc\hello.cpp	In function 'int main()':
4	5	D:\cc\hello.cpp	[Error] expected ';' before 'return'

行: 4 列: 5 已选择: 0 总行数: 6 插入 在 0 秒内完成解析

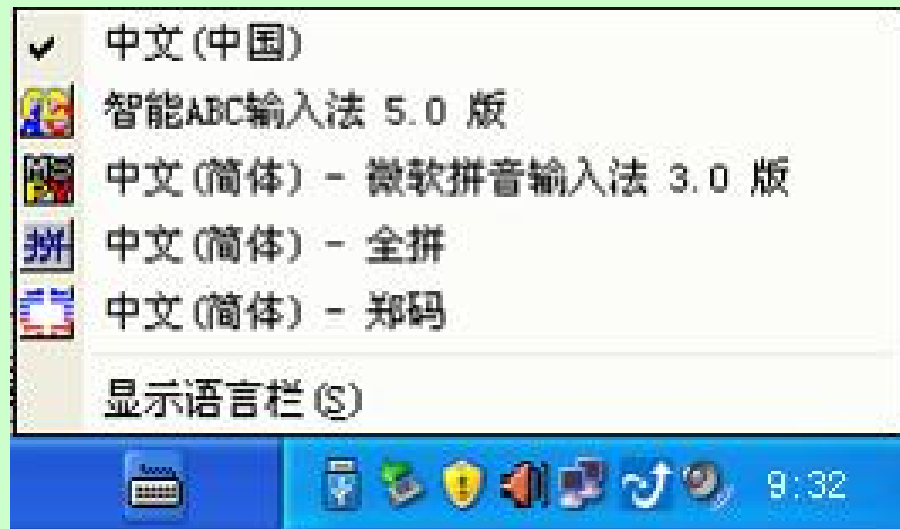
初学者容易编译出错的例子(4):



Windows中输入法的基本操作

- 切换输入法：

1. 鼠标切换



2. 键盘快捷键：

记住常用快捷键：

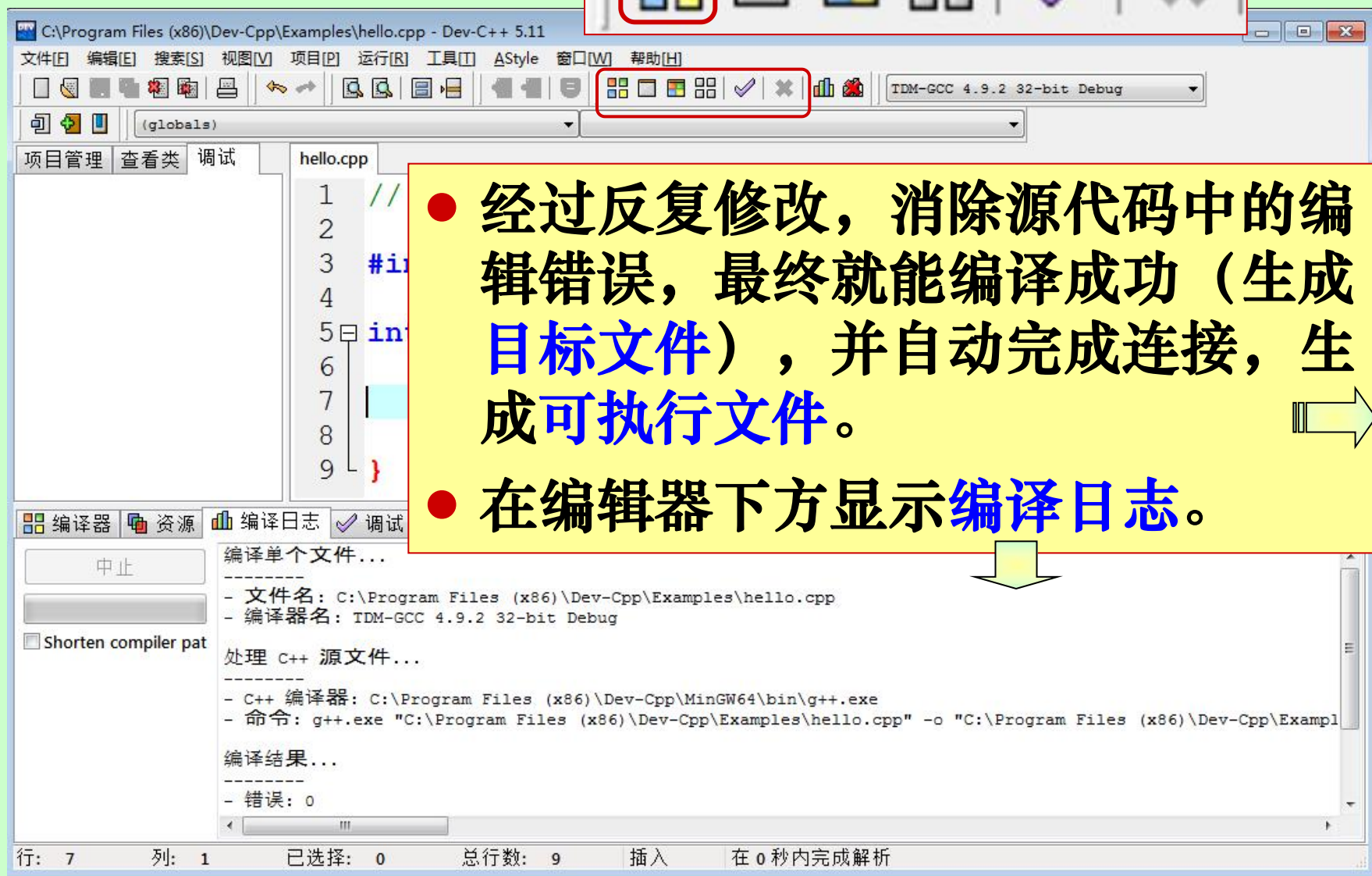
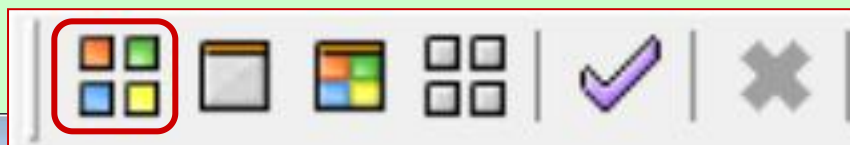
切换中英文：**Ctrl+空格 (XP)**

Win+空格 (Win 7/8/10)

切换中文输入法：**Ctrl+Shift**

切换中英文标点：**Ctrl+ .**

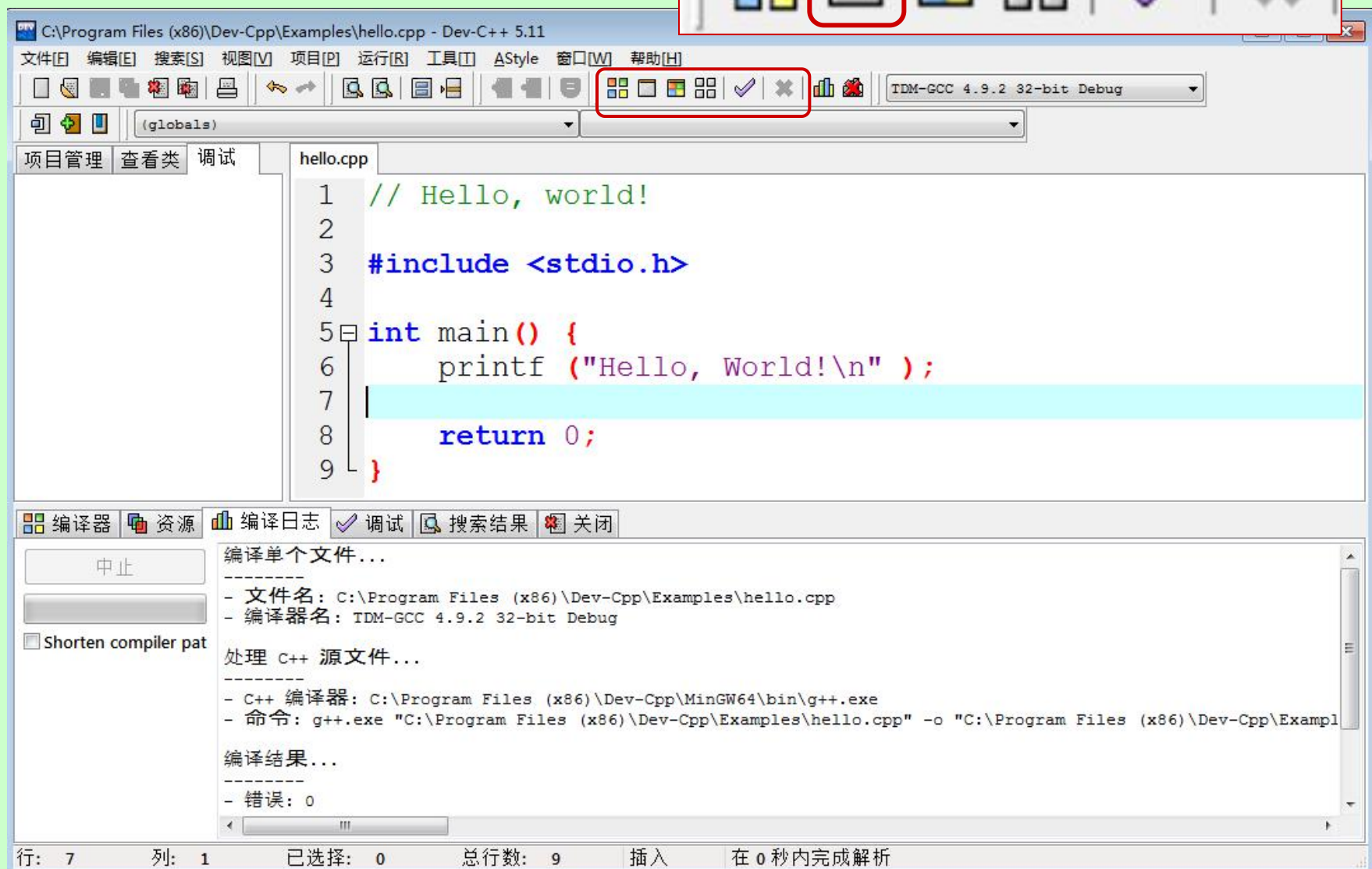
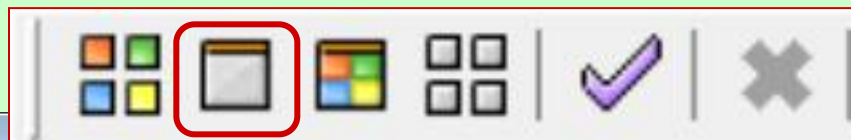
(按住 Ctrl 键不放，再按其它键)



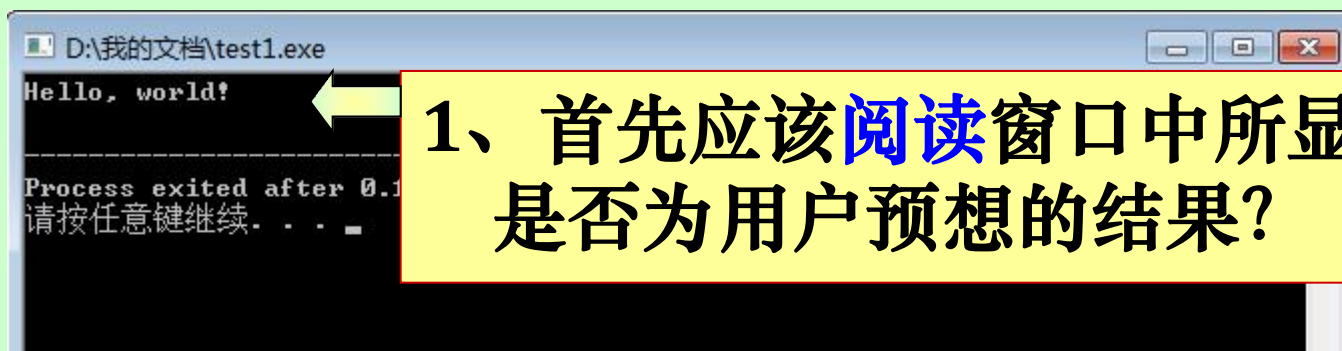
在文件管理器中可以看到这些文件：



- 编译连接成功之后，点击“运行”按钮就可以运行所生成的可执行文件



文字界面程序运行时会出现 控制台窗口：



1、首先应该**阅读**窗口中所显示的信息，是否为用户预想的结果？

2、程序正常结束时，**应该按 回车键 以便让该窗口正常关闭。**

如果未正常关闭，可能在以后编译时出现奇怪而难以处理的问题。

Cannot open output file ***.exe: Permission denied
[Error] ld returned 1 exit status**

C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\bin\ld.exe
C:\Users\...\Desktop\collect2.exe

cannot open output file C:\Users\...\Desktop\未命名1.exe: Permission denied
[Error] ld returned 1 exit status

- 学习了上述操作之后，请说出你如何理解下面这句话：**C/C++ 是高级程序设计语言，其开发过程要经过编辑、编译、连接和运行这四个步骤。**
- 解释以下名词：**集成开发环境 编译器
源程序文件 目标文件 可执行文件**
- 请在文件管理器中找出以下各类型的文件：
源程序文件 目标文件 可执行文件

练习题

1. C 语言是一种_____化程序设计语言。
2. C 语言有两个主要的技术标准： _____和 _____
3. 用户编写的 C 语言程序文件称为源文件。C 语言源文件的扩展名是_____，也可以按照 C++ 语言的规则写为_____。
4. C 语言源程序需要经过_____（扩展名为 _____）和 _____（扩展名为 _____）才能生成可执行文件（扩展名为 _____）

5. C 程序中语句必须以_____作为结束标记。
6. 一个C 语言程序是从_____函数开始执行的。
- 7.构成 C 语言源程序的基本单位是（）。
- A.子程序 B.过程
- C.文本 D.函数

8. 在一个 C 程序中，main() 函数（）。

- A. 必须出现在所有函数之前
- B. 可以在任何地方出现
- C. 必须出现在所有函数之后
- D. 以上都不对

9. 下列叙述中正确的是（）。

- A. C 程序的执行是从 main() 函数开始，到本程序的 main() 函数结束
- B. 注释语句是必不可少的
- C. main() 函数必须放在其他函数之前
- D. printf() 是系统提供的输出语句

1.3 C++ 程序快速入门

```
#include <stdio.h>

int main () {
    printf("Good morning!\n");
    return 0;
}
```

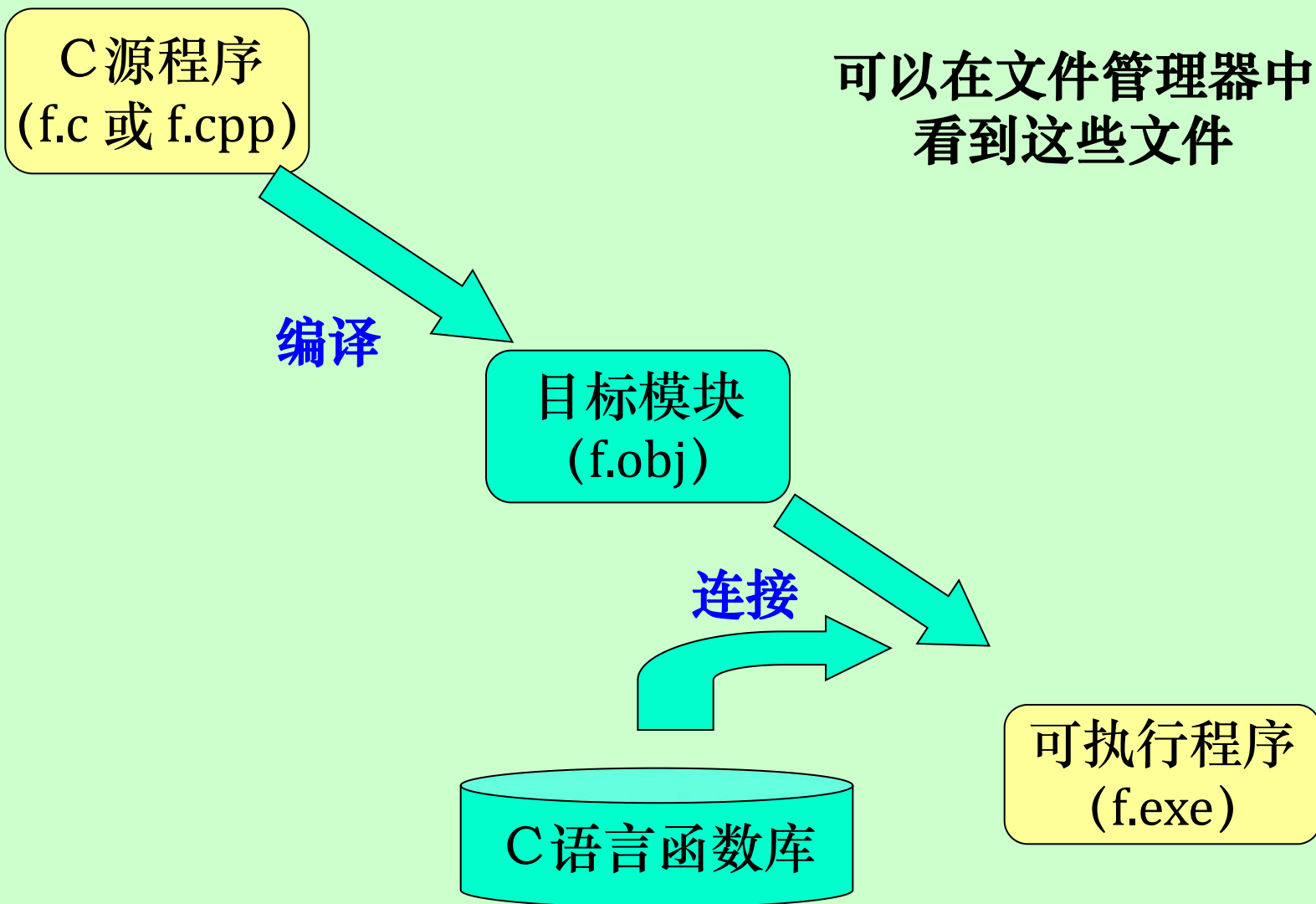
- 注意程序基本部分的写法。
- 程序是字符序列，应该按照易读的形式分行。程序形式应较好反映程序本身的层次结构。
- 可用普通编辑器编写程序，现在人们常用专门的程序开发系统写程序。

程序格式

- C语言是自由格式语言，可随意安排格式（换行/加空格等），格式变化不影响程序意义。
- 程序需要给人看，可能很长，结构可能复杂。
- 格式应很好体现层次结构，反映内在关系。规则：
 - ◆ 适当加入空行，分隔不同部分
 - ◆ 同层对齐，下层退格（加空格/制表符）
 - ◆ 加必要的说明信息（后面介绍）
- 写简单程序时就应注意养成好习惯。自由格式语言使人能根据需求和习惯写出格式良好的程序。

C 程序的加工和执行

- 用户按照 C 语法规则写出的程序称为源程序，不能直接执行。
- 为执行 C 程序，必须先把它转换为可执行程序。这种转换称为 C 程序的加工，是 C 语言系统的主要功能。
- C 程序加工通常分两步（见下页图）：
- **编译（Compile）**：编译程序处理源程序，生成机器语言目标模块，目标文件。目标模块不能执行，缺少必要的**C程序运行系统**和库功能。
- **连接（Link）**：连接程序把**目标模块**与**运行系统、库模块**组合起来，构成完整的可执行程序。



可以在文件管理器中
看到这些文件

C 程序加工过程

上面的程序执行输出一行，显示在屏幕窗口中：

Good morning!

- 不同C系统启动程序加工的方式不同。
- **集成程序开发环境**（Integrated Development Environment, **IDE**）是支持软件开发过程的软件系统。
- IDE把编程所需软件集成起来统一管理和使用。采用窗口菜单技术，提供编程用编辑环境，通过菜单提供编译、连接、执行程序等命令。
- 具体 IDE 的操作方式可能有些差异，但它们的基本功能相同。学习一个就可以触类旁通。

- Dev-C++ 对初学者是最方便的，安装和使用都很简单。
- Code::Blocks 在安装时比较麻烦，需要分别安装编译器、Code::Blocks 以及中文汉化包。使用时倒与 Dev-C++差不多。
- 微软公司的 Visual Studio 系列是商业软件，通常只能用盗版。功能特别强大，而且都是基于工程进行处理的，对初学者来说开始使用时比较复杂。

1.4 集成开发环境 Dev-C++ 使用简介

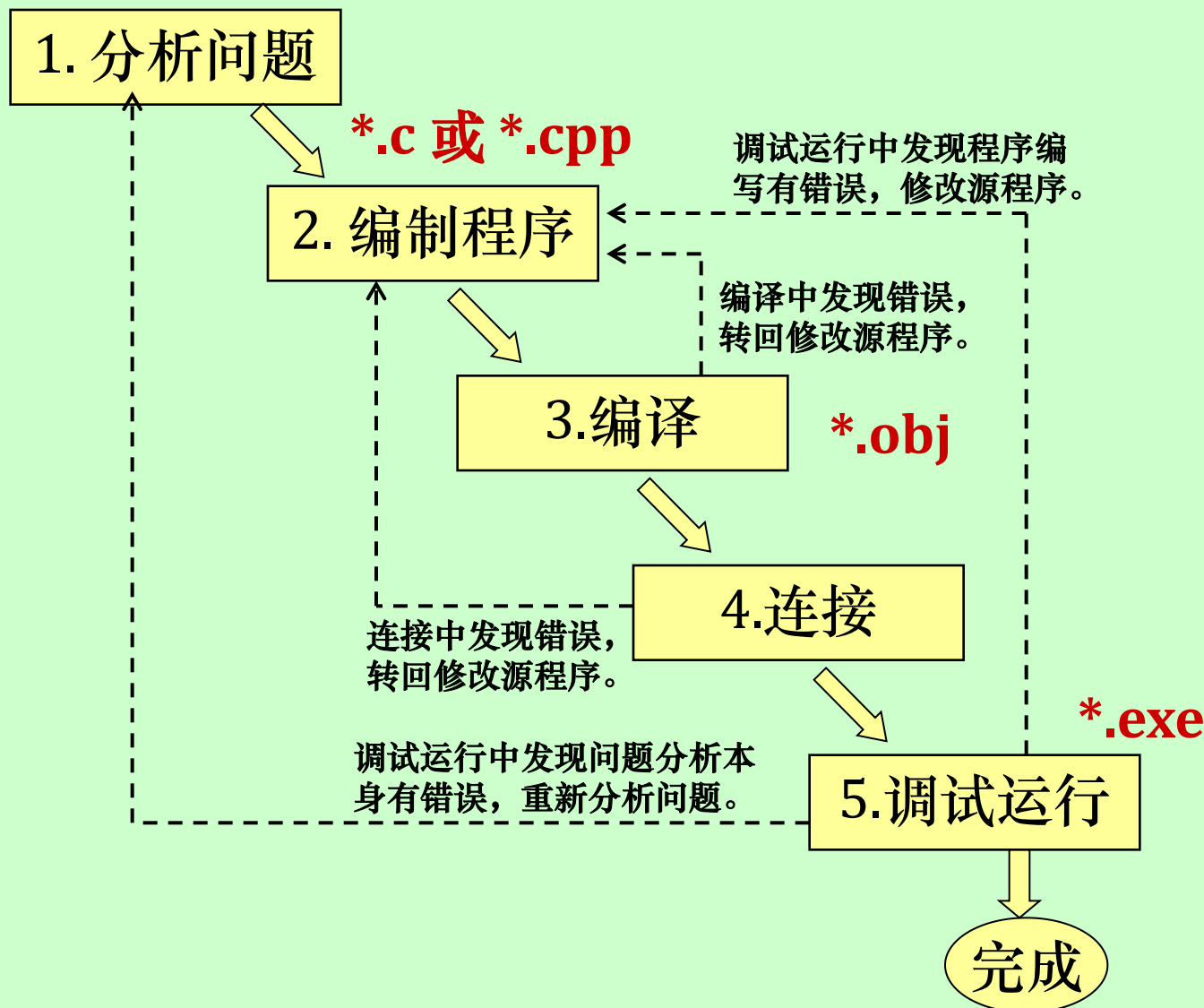


图1.3 程序的开发过程

- **调试 (Testing) 和排除错误 (排错, Debugging)** 是编程的必经阶段。简单介绍:
- **程序中的错误是人的错误**。排除程序错误就是排除自己在程序设计中
所犯错误, 消除自己写在程序里的错误。
- 错误可分为两类:
 - 1) 程序形式不符合语言规定。C语言系统在加工时能指出程序里的这类错误。
 - 2) 程序形式正确, 能完成加工, 产生可执行程序。但程序工作不正常: 或在执行中出问题, 或计算结果 (或执行效果) 不合要求。

- 语言系统查出错误时，将产生一些“错误信息”行，指明发现位置和错误类型，供人参考。
- 注意：应仔细阅读系统报错信息，检查所指位置附近的源程序，找到实际错误并予以排除。
- 基本原则：集中精力排除系统发现的第一个错误。
- 两个问题：
 1. 实际错误可能出现在指定错误位置前面很远处。
 2. 一个实际错误有时会导致许多出错信息行。
- 应注意警告信息（WARNING）：警告常表示隐藏较深的错误，必须认真弄清原因。

- 排错示例：《快速入门》PPT中介绍了初学者常见的错误，包括缺失字符、非法汉字等。
- 详细的调试方法将在以后介绍。

1.5 问题与程序设计

- 程序设计是智力劳动，编一个程序就是解决一个问题。
- 用计算机解决问题的过程可分为三步：
 - 1) 分析问题，设计一种解决问题方案；
 - 2) 用某种程序语言严格描述这一解决方案；
 - 3) 在计算机上试用程序，看它能否解决问题。
- 在运行时发现错误，就需仔细分析错误原因，退回到前面步骤去纠正错误，直至得到满意结果。
- 第一步中需要从计算和程序的观点出发，也带来了许多新问题。第二、三步是程序设计的特殊问题。

本课程学习中应注意的问题

1. 分析问题的能力，特别是从计算和程序的角度。
2. 掌握所用语言，熟悉语言的各种规定，形式和意义。
3. 学会写程序。

解决同样问题，写的程序是否较简单？是否采用了合适的结构？是否清楚、易于阅读和理解？一些条件改变时，程序是否容易修改，以满足新要求？等等。

4. 检查程序错误的能力。确认实际错误，弄清楚应当如何改正，这永远是编程序的人的事。
5. 熟悉所使用的工具和环境。

上机文件命名规则详解

例2.1

学号后四位-姓名缩写-ex2-1.cpp

习题 2.1

学号后四位-姓名缩写-xt2-1.cpp

不要手工写 .cpp，
它会自动加上的。

要把点都改成横杠，以便使系统能自动添加扩展名。

(否则系统会误以为 “.1” 是用户指定的扩展名呢。)